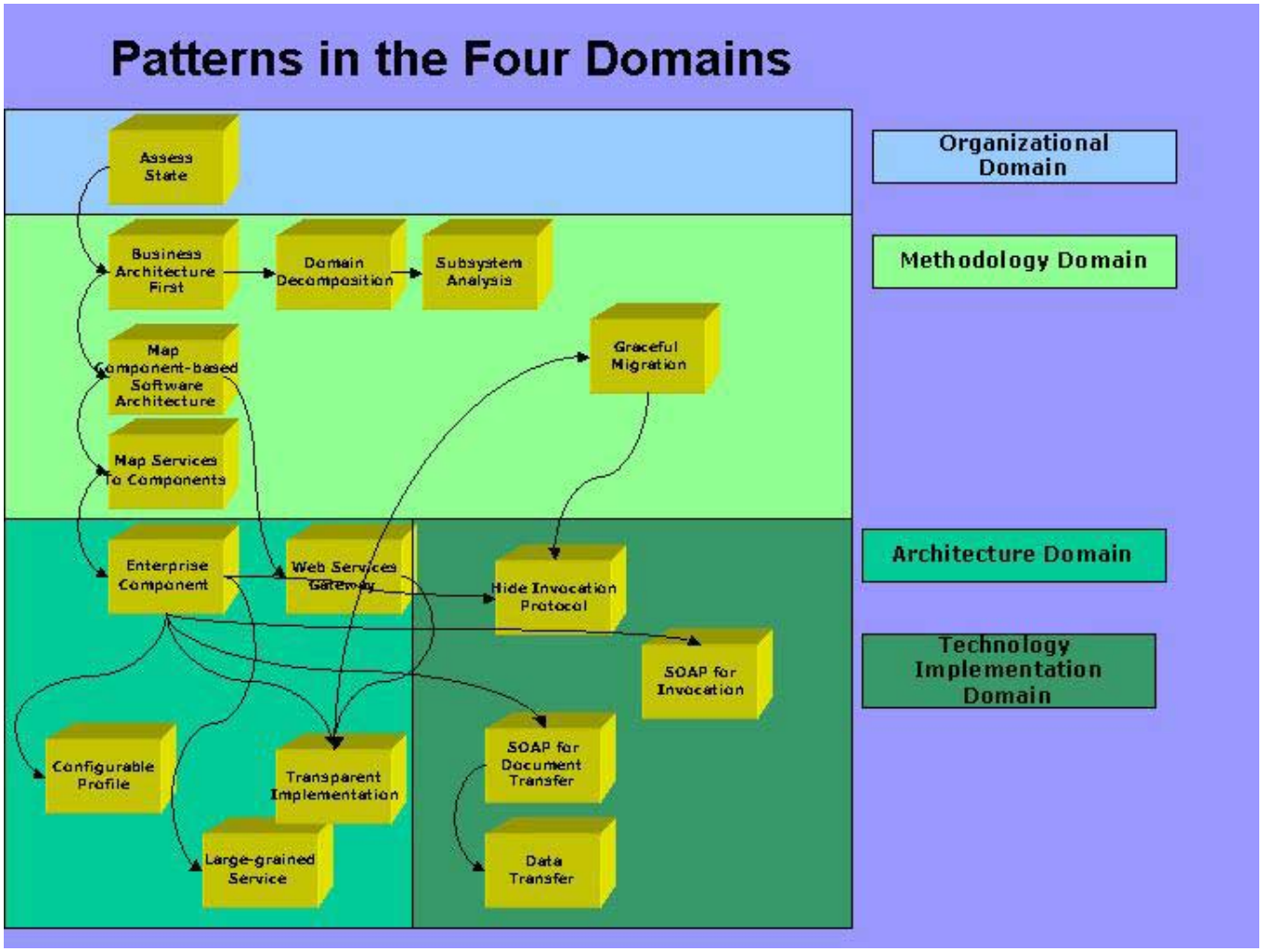
Towards a Pattern Language for Web Services Architecture

Ali Arsanjani,
Senior Consulting I/T Architect, Component Competency Lead
IBM Global Services National EAD Center of Competency and
Maharishi University of Management
arsanjan@us.ibm.com

1.1 Introduction

The current set patterns include both methodology and architecture patterns. The pattern language consists of patterns for five levels or domains: organizational, methodology, architecture, technology implementation and infrastructure. In this paper, we will cover the first four domains. It is necessary not to jump into technology at the start but use these patterns to migrate the organization, methods and architectural decisions along the right path before applying the more technology focused implementation mechanisms. It is with this context that we present the first set of patterns for Web Services Architecture.

1.2 A Map of the Patterns



Discussion. The above RoadMap shows that there are four domains in Web Services. In each domain, there are corresponding patterns that apply to resolve forces that arise in that domain. But the application of the patterns may result in forces being unbalanced in other domains; for example. Map Component-based Software Architecture uses Web Services Gateway and Enterprise Component, which are Architecture level patterns.

The Patterns

Below you will find the detailed description of each of the patterns in the following format:

Context	The background, context and situation in which the problem arises and then forces have tension.
Problem	The issues and problems that arise in the context often as a result of conflicting forces that “pull” in “opposite” directions and call for design decisions that balance forces in the solution.
Forces	The set of often antagonistic and perhaps mutually exclusive elements that decision need to be made on. These “parameters” of the problem space are resolved by making tradeoffs in given contexts to resolve the forces.
Solution	The solution is a resolution of the conflict by balancing forces based on design decisions that a master designer makes in a given context. If the context changes, perhaps other (different) design decisions would have been/will be made.
Diagram	A pictorial depiction of the solution, often with elements of the problem evident in it.
Solution Detail/Discussion	If necessary a discussion of the solution, its details and implications.
Resulting Context	What is the result of applying the solution to try and balance the forces? Do they all get balanced or do some remain imbalanced? Does the introduction of the solution lead to the unbalancing of further forces that other related patterns in the language are to try and balance out?
Implementation Considerations	Where applicable
Known Uses	These have been project the author and colleagues within IBM have been working on for the past few years. Other industry reports and projects point in the same directions, but no explicit reference can now be stated other than domain or industry in which the projects were implemented.

2 Assess Current State: Assess Current Integration Level

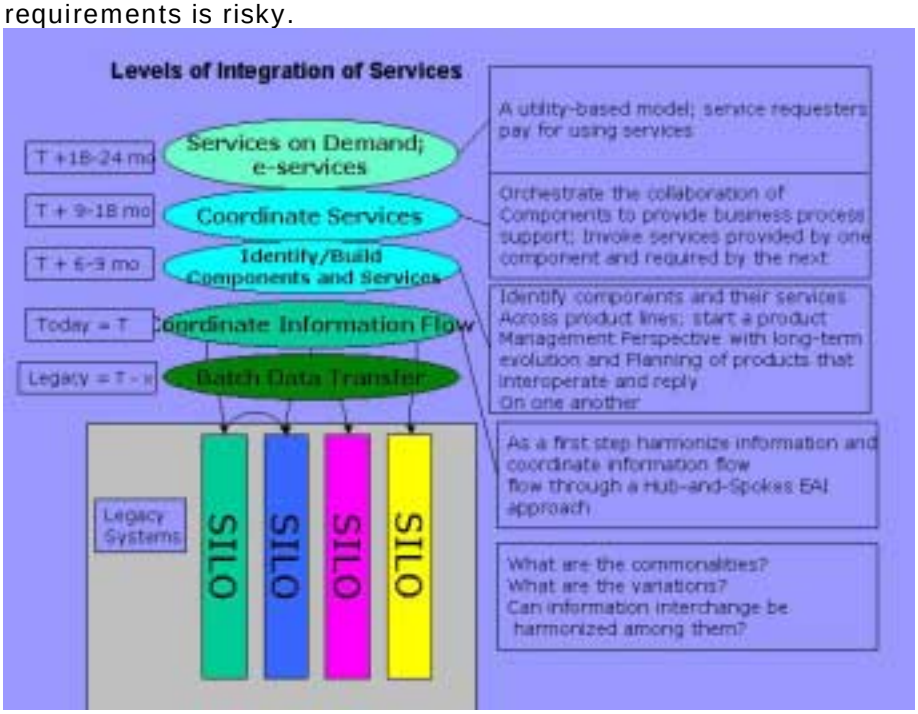
Context — Organizations are at various stages of architectural maturity. They need to move on to the next stage in their development.

Problem — Should we adopt Web Services immediately? If not, how can we know when we are ready to migrate? What is the migration path to web services or a service-oriented architecture?

The problem can be expressed as the inability of the organization to determine whether it is in terms of information integration; is it mature enough to start with web services or should it really start by creating an EAI infrastructure and /or a component-based architecture first before randomly applying technology to expose web services.

Forces — You need to stay ahead of the industry to provide competitive edge and yet technology that is still immature or has some pieces missing in terms of performance, security and other non functional requirements is risky.

Diagram



Solution — Understand the levels of integration and corresponding enterprise architectures, study the organization and identify where the organization is today (often various business lines will have different levels of maturity or concerns) and plan to get to the next level by addressing organizational, methodology, architecture, technology implementation and infrastructure/tools issues.

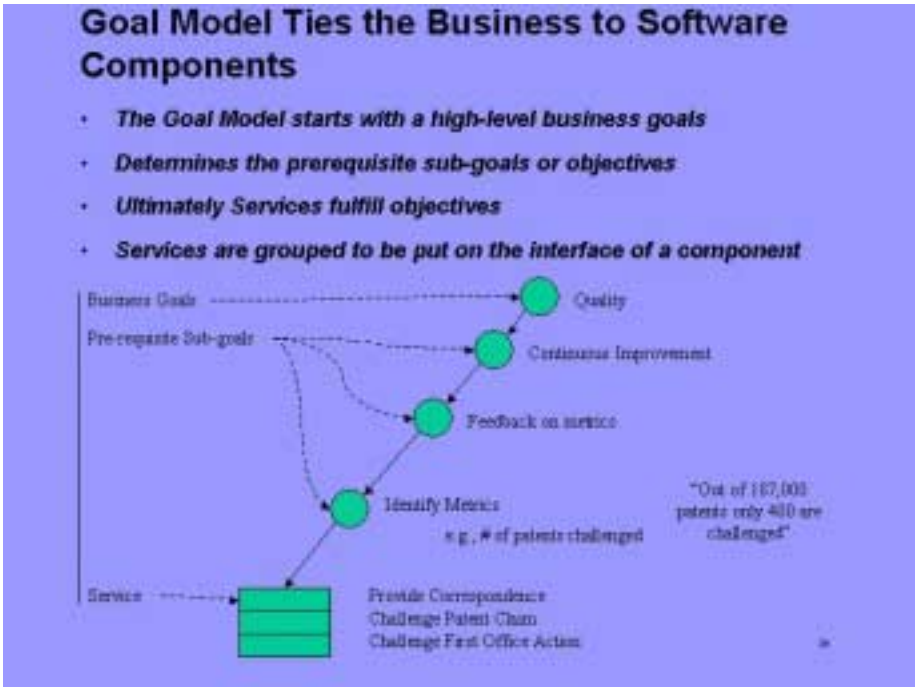
Solution Discussion — It is key to identify where the organization is in terms of architecture before embarking on the voyage of a service-oriented one. There are steps of maturity and levels of integration that build on top of one another. It is not recommended to “jump” from one level to the other as a lower level provides the infrastructure and means to create and support the next level.

Resulting Context	You have now planned and are executing your migration strategy based on where you are today.
Implementation Considerations	You may not need to go all the way up to a utility-based model of services; a component-based level may be sufficient for satisfying the business goals of your organization.
Known Uses	Various projects done in telecommunications, banking, mortgage and credit card sectors.

3 Build Business Architecture First: Business Architecture Drives Software Architecture

Context	Business drivers and technology constraints are different. They different tools and create often different models. Requirements handed down from business to I/T often suffers because of this conceptual mismatch, ambiguity and model gap. Technology is often not aligned with the business and does not deliver value by specifically addressing business goals.
Problem	What kind of architecture will lead and drive the other: business or I/T? What are the considerations? Business requirements are usually vague and much is left to interpretation by I/T. This gap widens as services are exposed that may not make business sense to do so. Therefore, the problem is how to use business drivers to define services and components?
Forces	Should we start building the I/T infrastructure first? Should the business drive this process? Can business goals and requirements drive software or is software architecture merely an implementation to be made that will fulfill the needs of the business without the need for explicit mapping from one to the other?
Solution	Therefore, build the business architecture first. Create a Goal Model, Process Model, Rule Model and Conceptual Model of how the business operates within the scope of the domain under study.

Diagram



Solution Discussion

It is necessary to ensure the business value of exposed Web Services. This is partially accomplished by ensuring traceability of web services back to business processes and goals. To accomplish this, we use the Goal Model, which is represented by a Goal-Services-Graph. This model defines higher level business goals and gradually refines them using sub-goals until objectives/sub-goals can be realized using one or more services required to fulfill the goal. Thus, at each level we could have a set of services associated with a sub-goal. A convenient alternative is to have the services only at the leaf node level of the goal graph (as depicted above, in the structure of the solution).

Use the goal model as a mechanism to map the services that the software architecture must provide onto the business model and its goals; thus tying software architecture back to business goals.

Often the task of choosing large-grained component boundaries is a difficult task. Therefore, Goals Define Services.

Resulting Context

You now have a business architecture as a driving force behind defining the software architecture. The Enterprise Components of the software architecture provide services that map back to and promise fulfillment of business goals in a highly traceable fashion.

Implementation Considerations

Take care to list out all services that are required to fulfill a sub-goal.

Known Uses

Large financial services company for internal and international divisions, Large mortgage company for institutional loan processing.

4 Build Component-based Software Architecture

Context — You have built a Business Architecture and have defined the goals of the business and have mapped services to goals. You now want to go ahead with the details of defining your component software architecture. You want components that will expose services to business partners and customers.

Problem — How do you define component boundaries? How do you decompose the business into large-grained enterprise components? What do you expose as services to the world at large: customers and business partners?

What criteria do you use for domain partitioning or decomposition?

Services cannot be exposed on their own without components to provide infrastructure. Otherwise the services become difficult to maintain, test and track, from a change management perspective (see Services Map To Components). But components have to be chosen carefully. Component identification and specification is a major step in mapping from a business to a component-based software architecture.

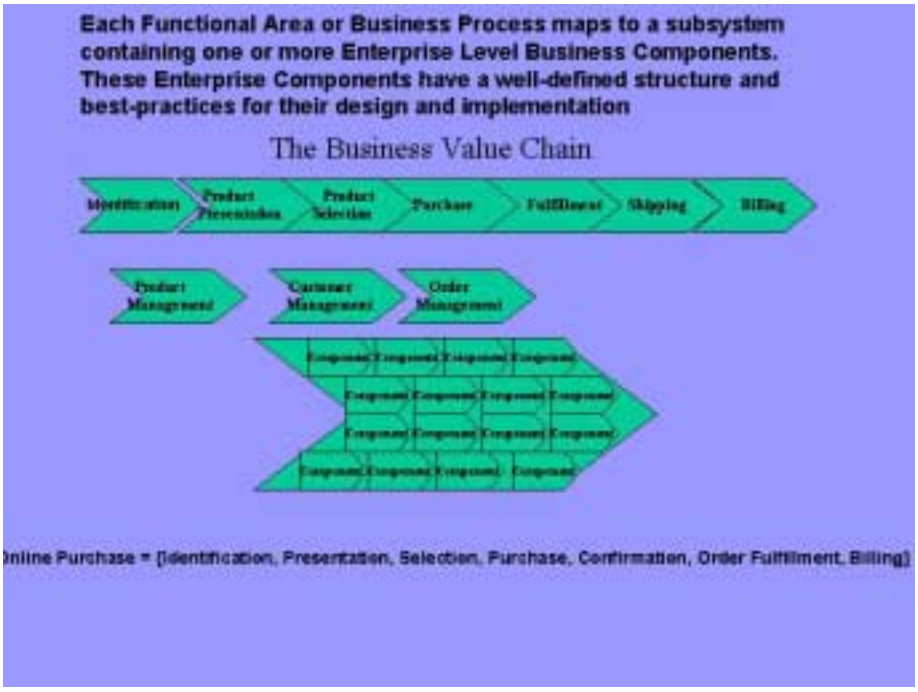
Forces — Do you expose fine-grained services, such as methods on a class, or do you expose large-grained business process level services?

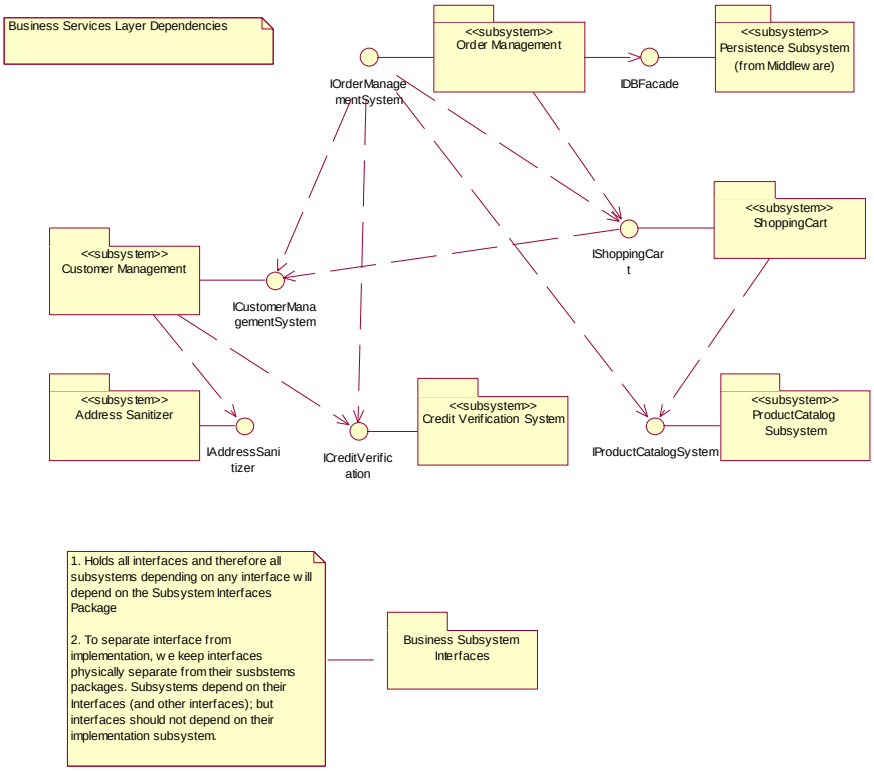
What business objects are in a component's jurisdiction or boundary and what falls outside: there are many ways to decompose – which set of criteria should you use for large-grained component architectures?

You can decompose by functional area or business process or you can partition into business objects causing more interaction and back-and-forth "chatter" between elements in the business that will be supported by the software architecture.

Solution — Processes and functional areas provide a good large-grained criterion for domain decomposition. Decompose the business domain into a set of large-grained Enterprise Components by mapping high-level business processes to subsystems. Expose services on those enterprise-scale components that correspond to business process level functions.

Diagram





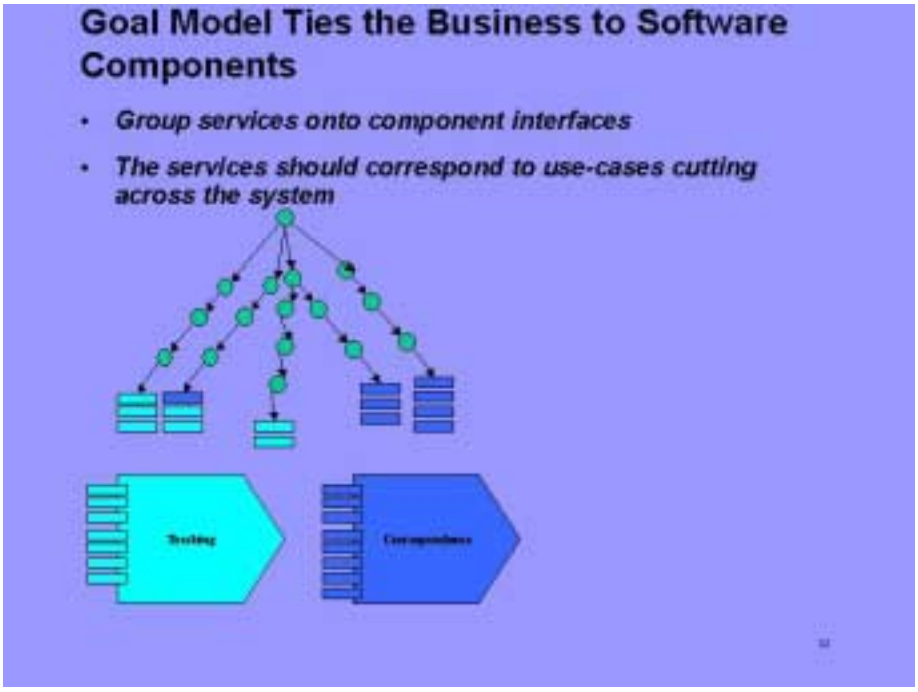
Solution Detail

The domain is partitioned into a set of business process level units of functionality that align themselves with business goals. Dependencies between the large-grained components or Enterprise Components are explicit and help you define the sequence of development.

	Why do we need to build components in the first place? Because services must ride on top of modularized chunks of business functionality: enterprise-scale component.
Resulting Context	You have a software architecture built around large-grained enterprise components that expose services that map back to the business goals. The business process areas or functional areas of the business are supported by and mapped onto loosely coupled Enterprise Components.
Implementation Considerations	The management of related functions under a business process point to including them in the same subsystem. Often the data services for a component should be managed by the component itself. Don't forget legacy integration and transformation. You can capitalize on legacy functionality by componentizing it and including it in the software architecture component model. The large-grained components are themselves a Composite and consist of medium to fine-grained components or business objects or links to legacy systems (see Enterprise Component).
Known Uses	Parnas suggest encapsulating design decisions and in the business realm this relates back to business processes and the goals and rules within them.
Related Patterns	Services Map To Components; without a component-based infrastructure, Services cannot be exposed on their own without components to provide infrastructure. Otherwise the services become difficult to maintain, test and track, from a change management perspective.

5 Services Map to Components

Context	You have defined a software architecture and its enterprise components that map to business architecture, processes and goals. But you are still not sure the component boundaries or services are complete or all correct. Often it is unclear what level of granularity to express as a Web Service.
Problem	How do you verify functional coverage of the entire domain and validate completeness of the software architecture as far as it provides enough and correct services that map to business goals?
Forces	Methods can be implemented by different components? Which component does a given function or service belong to? It can belong to two or many : which one should you choose as the implementer or provider of the service? Clearly fine-grained components should not be exposed to the external world; only business level services that provide value should be exposed.
Solution	Use the Goal Model (goals and sub-goals and services) as a guide to cluster services. Assign services to Enterprise Components based on the functional areas they relate to and the goals they fulfill.

Diagram	
Solution Detail	These services correspond to the use-cases that are encapsulated within the enterprise component.
Resulting Context	Enterprise Components in your software architecture now have the set of services that are required to support the business. This includes a set of Internal and External Services[1]
Known Uses	Loan processing system exposes services that rely on underlying components for handling them. Large financial institution relies on exposure of services within the enterprise in a loosely-coupled fashion but have large-grained components as the underlying infrastructure.

[1] See pattern with same name.

6 Enterprise Component

Context — You have defined the software component architecture and are sure that all Services Map to Components based on the Goal Model. The next step is to define the internal workings of this business process level, large-grained subsystem-level component.

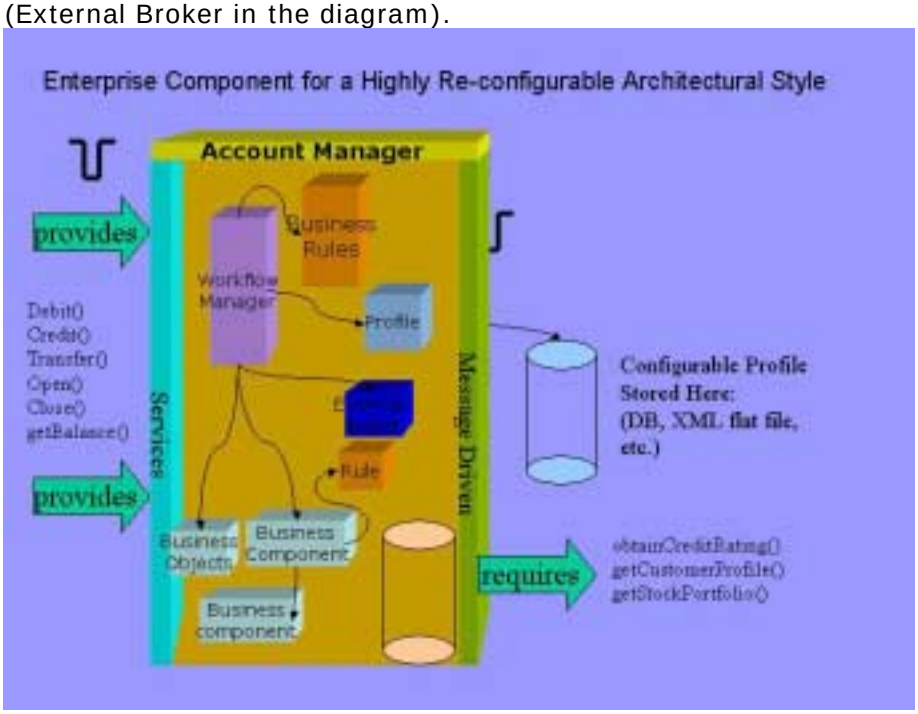
Problem — How do we construct these enterprise-scale large-grained components? What is their internal structure and function?

Forces — Each development team can build its own version of a large component or should we standardize across the organization?

Should we allow teams to build their own internal designs for large-grained components? Or are standards like EJB enough to define the internal structure of components? Perhaps not.

Solution — Therefore, create an Enterprise Component out of five main patterns: façade, Mediator, Composite, Rule Object and Adaptor (External Broker in the diagram).

Diagram



Solution Discussion — Use a Façade to encapsulate design decisions within the subsystem that maps to the business process. One or more mediators inside the EC will handle groups of related service requests for load balancing and separation of concern purposes. Business rules across three levels of granularity are needed to encapsulate and capture the changes in business rules for the Enterprise Component, medium grained components that make up the Composite Enterprise Component and the leaf –level business objects or legacy level adaptors that need to be accessed to actually provide a given service or group of services.

Resulting Context — You can now teach this compound pattern to team leads and have a standardized way of designing and implementing EC's within the business line and across the organization.

Implementation Considerations	Each individual element or participant within the EC can be built separately facilitating team development and distribution of effort/labor. Some can focus on legacy integration while other parts of the teams or sub-teams can write the business objects and rules. An important aspect of the enterprise component is that it should implement messaging, SOAP and RMI-IIOP to be able to handle all invocations through these access mechanisms and protocols.
Known Uses	Many projects in telecommunications, banking, mortgage, patent processing, etc., use this paradigm in their component models and implementations.

7 Large Grained Service

Context	You have defined and constructed components. Now you want to decide which services to expose to whom. But first you need to solve the problem of service granularity for business partners and customers.
Problem	Should you expose a fine-grained method on a class or a larger grained business –level service?
Forces	Should we provide a high-level service as would trigger a business process or should we provide lower level access to internal I/T capabilities? What are the intellectual property implications of exposing lower level finer-grained services to potential competitors?
Solution	Therefore, only expose business processes or high-level use-cases through the services to clients or business partners.
Diagram	
Solution Discussion	Use-cases often are broken down two or three levels of detail into sub-uses and "sub-sub" use-cases. Higher-level use-cases are often mapped back to high-level business processes. These processes directly support the business while the lower level ones need not be exposed and only serve to fulfill higher-level ones.
Resulting Context	You have a set of Enterprise Components that expose large-grained services to business partners and clients.
Implementation Considerations	The level of granularity of the service depends on the Goal model created; the service of business value (satisfies goals) should be exposed based on a decomposition of business process.
Known Uses	Exposure of credit card services for members and third parties in a large financial institution.

8 Use SOAP for Document Transfer

Context	You have identified the data transfer points between components within the enterprise or across the extended enterprise.
Problem	How do you transfer information between arbitrary nodes in a distributed computing environment where components are hidden behind firewalls?
Forces	You can use any XML stream to send and receive information; and yet standard envelopes and protocols seem to be better handled by tools, better recognized by other business lines or partner organizations. Should you opt for a standard protocol or "roll your own"? Each have their merits and drawbacks. But you want to also start getting web services into the organizational culture.
Solution	Therefore, define the content of messages inside the SOAP content and use its header as a standard. Transfer the data using the SOAP envelope. The data will be represented in XML format but would be packaged in a recognized SOAP-based envelope format.
Solution Detail	You are not constrained to use SOAP for invocation and can use this as a "foot-in-the-door" to get the enterprise thinking in terms of using Web Services in a familiar way and "break-in easy".
Resulting Context	Data transfer with regular XML is possible; but when you would like to evolve into RPC, you want to avoid boxing yourself into a corner.
Implementation Considerations	Using a SOAP header to transfer data in XML over HTTP can be a first step to using Web Services protocols in your organization, without exposing you to the risk of lower performance by using SOAP. So instead of merely sending XML, you put the XML inside the SOAP envelope.
Known Uses	International Electronic Patent Application Submission Credit Card Company information interchange between third party vendors and credit card services

9 Use SOAP for RPC

Context	You have set up the organization to use SOAP for Data Transfer. People have learned how to use SOAP and how to transfer data. You have defined Large-grained Services and have Tested Them Inside and are now wanting to Apply Services Outside.
Problem	You need loosely coupled services invocable across the internet, from anywhere. How do you expose services that are invocable over basic internet protocols?
Forces	You need to expose services worldwide to clients and vendors; yet you need to maintain control over the implementation of these services. You can mandate protocols, but most loosely coupled systems cannot abide by more stringent CORBA-like protocols over the extended Internet.
Solution	Invoke remote loosely coupled services using SOAP over HTTP or HTTPS (as the need for security changes)
Solution Detail	The implementation detail in the backend could bind to any kind of technology using a Web Services Invocation Framework-like

	protocol.
Resulting Context	You now have a set of business level, large grained services exposed for invocation throughout the enterprise and a subset are exposed to customers and business partners.
Implementation Considerations	The SOAP envelope consists of a header and a body. The body can contain any valid XML-based content. Use the header for protocol recognition; use the contents to send MQ messages containing XML content models. Valid the content models upon reception and before further processing. Separate out the content validation from the back end business processing (See Separate Validation from Processing pattern)
Known Uses	Expose services for vendors in a supply chain for major credit card company, third party vendors and other product lines; Expose Services for Third-parties to use Patent Application information; Expose Major Mortgage company's loan application services as invocation-based services

10 Self-describing Service

Context	You have decided which Large-grained Services will be exposed and implemented by which Enterprise Components. Now you want to actually define the interfaces.
Problem	What protocol or interface should you use?
Forces	You may want to Use SOAP for Data Transfer or Use Soap for Invocation. But you may not want to bind the implementation to use SOAP exclusively inside your firewall. But you may want to migrate to any given technology for invoking the services.
Solution	Describe service in WSDL providing self-description features.
Solution Detail	Later, you can use Web Services Invocation Framework or a Web Services Gateway to bind the implementation to legacy systems, message-queue based systems, EJB's or CORBA. But you are covering yourself for UDDI and cultural migration by defining all services in WSDL; whether you want to use SOAP or not.
Resulting Context	Now you have to worry about non-functional requirements. These are specified in a Configurable Profile and is bound to implementation through a Transparent Binding.
Implementation Considerations	Use an internal UDDI registry to store /find/bind internal company confidential information; use an external UDDI registry to expose less competitive information accessible to clients and business partners. The two registries is an important implement consideration that allows a gradual migration (See Graceful Migration) from secure, intellectual property-based services that provide competitive edge to those that are okay to be exposed to business partners and vendors.
Known Uses	Used for business lines in a large financial organization to gain access to a suite of previously inaccessible legacy services that are wrapped and exposed as web services for inter-company usage across product lines and business lines.

11 Configurable Profile

Context You would like to be able to configure the software system based on the needs of different users. I imparts this is a form of personalization. Each user time may have a personalized configuration, which is initialize whenever the user logs onto the system or attempts to utilize a specific service. Often within the context of business lines in product lines, we encounter the situation in which a large company consists of a set of subsidiaries. Each subsidiary will have its own unique way of doing business along the same general guidelines as its parent company. In order to develop applications software for each business line or product line, it is often necessary to rewrite portions of the application based on a particular language, currency or other internationalization characteristics. Another type of customization or personalization, pertains to that of security. Each user within a particular geographical location for office branch, may have a set of rules assigned to them. These rules can be encrypted an externalized as a configuration profile. They can also be stored in encrypted form, in a database management system.

Yet another form of personalization is that of a configurable profile. Each user type will have a profile of its own. For example, institutional users will have a different profile the retail users of the loan origination system.

In the telecommunications example, each product line such as Wireless, wireline, DSL, etc., may have a different set of initialization characteristics that define the way they conduct business by sitting parameters of workflow, defaults, reference data, Security roles, options, etc.

You want to be able to define a new profile for each user tied without having to make changes to an existing program. Therefore, often a set of options are externalized in the form of a properties file.

In the case of deploying Enterprise JavaBeans, the deployment descriptor is defined. This is often an XML file, which contains information about locations of files, database connections, transaction isolation and other properties. This is a configurable profile at the level of individual software components rather than the user type.

Problem How can we provide the ability to configure different user types or component types without making intrusive changes to program code?

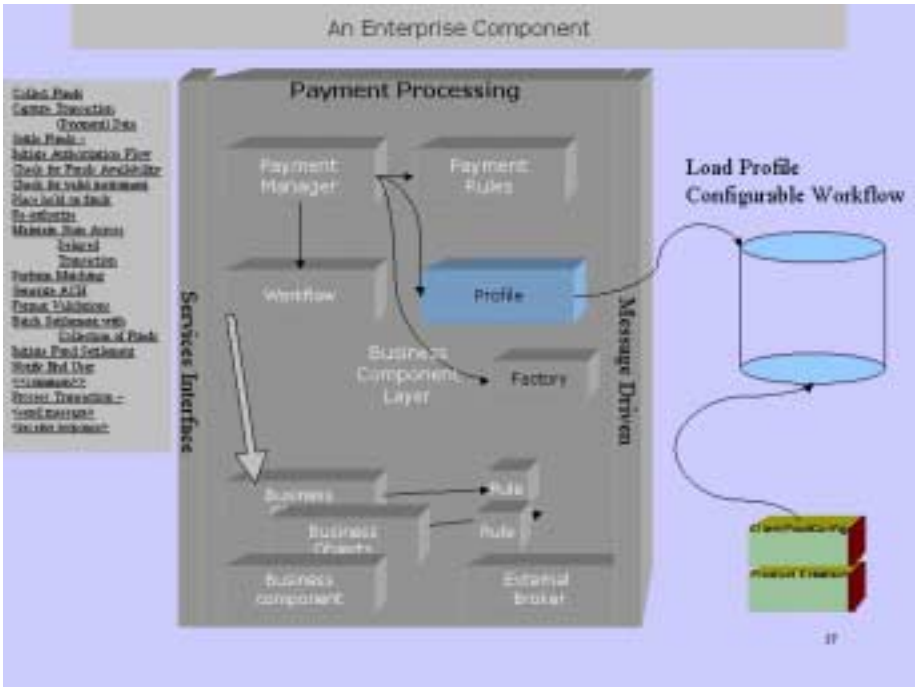
Forces Defined configuration parameters internally vs. externalization of configuration parameters: other general parameters that can be identified? Is it worth externalizing a set of configuration parameters; or the likely to be changed often to accommodate new types of users?

Complex configuration profile vs. simple configuration profile: should we attempted to their size a set of configuration profiles and attributes whose values will be assigned based on user type?

Solution Start with externalizing a set of obviously changing parameters based on user types. As the types of users change you can add new values to the attributes. You can also add new attribute sand combination of valid values as the need arises.

Therefore, externalize the personalization attributes of the user tied and create a configurable profile. Provide access to the configurable profile so that there will be no need to make direct modifications to program: in order to accommodate new types of users.

Diagram



Solution Detail

The structure of the Profile(s) can be simple properties files, XML files or a domain-specific language describing the flow of business within the large-grained component.

Resulting Context

Now you have a configurable profile. The flexibility inherent in the configuration is based on your assessment of the needs of future users. Thus, the flexibility in configuration is dependent on your predictions about future variations that are likely to come about.

You'll know how to manage the potential growth of the configurable profile. The configurable profile often has a language of its own, with its own grammar. Therefore management of the grammar may pose a challenge for users programmers. Alternatively, if you don't have to change the grammar a lot, for using the configurable profile is straightforward.

Implementation Considerations

The mechanism for implementing the Profile can be anything from an XML file to a full blown database. The decision to make the Configurable Profile centralized or distributed lies in how you want to deploy the corresponding components and whether you want to have a central point of control.

You might consider multiple profiles, one for each type of deployment (office/branch versus country; business rules profile versus office/branch profile).

Known Uses

Use of Configurable Profiles in a large mortgage company for defining the flow for given deployments for business partners; Use of this profile as a means of deployment in international locations with various business rules, internationalization criteria for a large financial service company.

12 Web Services Gateway

Context	You want to use Web Services to allow access to your internal service resources – those you have chosen to expose (See Selective Exposure (hide the internals, show the ones that are not compromising for competitive advantage) . You have your application servers for processing (for example) incoming servlets or JSP requests (equally valid for ASP scenarios). But you need to handle the requests for Web Services.
Problem	How do you support inbound requests for Web Services in your architecture? Is there a layer or server that is responsible for this? Or do you use standard Web Server technology?
Forces	Standard Web Servers are inadequate; yet you need new functionality.
Solution	Use a separate server middleware component that provides an intermediary gateway between Internet and intranet environments during Web service invocations. Include in it a model for the management of services. Use "Interceptors" to act on requests/responses that flow through the gateway.
Solution Detail	Web Services Gateway is a middleware component that provides an intermediary framework between Internet and intranet environments during Web service invocations. It includes a model for the management of services (deployment, undeployment, etc.) and "interceptors" (those pieces of code that act on requests/responses that flow through the gateway.) The gateway currently handles only incoming SOAP/HTTP requests (using either the Apache SOAP or Apache Axis engines), but support for more channels will be added in the future. At this time, requests passing through the gateway may be sent to a Java class, an EJB, or a SOAP server (including another gateway.)
Resulting Context	You now have a tier in the architecture responsible for handling the incoming and outgoing requests and responses coming to and originating from Web Services based protocols and implementations.
Implementation Considerations	The gateway builds upon WSDL4J and WSIF (Web Services Invocation Framework) for deployment and invocation. The Apache SOAP and Apache Axis engines provide the entry points into the gateway. WSIL4J (from the WSTK) is used to generate WS-Inspection documents that provide references to the WSDL documents of deployed services. A service is deployed to the gateway by deploying a WSDL file that describes how the gateway should access it. Interceptors (*WSIFInterceptors,* to be exact) may be deployed to the gateway to intercept incoming requests and outgoing responses. Requests to the gateway arrive via one of the SOAP engines, are translated into a WSIF message, are passed through any interceptors that are registered, and are then sent on to the service implementation. Responses follow the same path in reverse.
Known Uses	Apache Axis uses a Web Services Gateway .

13 Transparent Implementation: Indirect Binding; Hide Invocation Protocol

Context	You have set up a Gateway and Component-based Architecture, you have Expose Services and are now actively servicing requests from customers and business partners, vendors and clients alike. But you have back-end systems that you need to leverage to actually implement the functionality you have just exposed through Web Services.
Problem	How do we separate interface from implementation in web services?
Forces	Should we use SOAP; is SOAP ready for prime time? Should we link back to legacy?
Solution	Use Web Services Invocation Framework to decouple the implementation from the WSDL interface definition.
Solution Detail	You can now leverage any of your back-end systems, Application-server based or legacy to support the services you have exposed through Web Services.
Resulting Context	You now can link to legacy and reuse all your current assets and are not constrained to using SOAP for web services implementation.
Implementation Considerations	You need the adaptors to talk from your gateway to each of the implementations; i.e., you need a binding mechanism. WSIF is such a framework.
Known Uses	Apache Apex project, several client engagements in financial services, mortgage, etc.

14 Graceful Migration; Selective Exposure

Context	You know you want to expose services; you are not sure when they are mature enough. You have Mapped a Component-based Architecture, Defined Large-grained Services and Enterprise Components.
Problem	When should you expose your Web Services to your clients and business partners?
Forces	Should you expose a service to one project or all projects across business lines? Should you expose all services withon UDDI registry?
Solution	Use two UDDI registries (at least); one for inside the enterprise for internal cross project and business line services and one for gradual migration of tested and robust services to clients and business partners.
Solution Detail	The first UDDI registry is internal the second one external. The first inside one tests your services and when it makes sense from both a business and technology view to expose them, then gracefully migrate them from the inside UDDi registry to the external public one.
Resulting Context	You now have preserved intellectual capital and competitive edge by exposing only those services that you want and make business sense and do not provide the competition with greater advantage. Thus you have not only staged your services, tested them, but have preserved your company's assets.

Implementation Considerations	The first internal one should be secure, the second one is by its very nature, exposed to the world at large. Have a strict and strong line of security between the two, on two branches of a network.
Known Uses	Many 'Back Office' and message vendors, such as Tibco, Siebel and SAP, have announced that they will be supporting the standards based Web service protocol stack. This will increase the ease by which internal processes can be exposed and woven into new business requirements across the value chain.

15 Separate Validation from Back-end Processing

Context	You have established SOAP as a data transmission vehicle. You are receiving content models and would like to process them.
Problem	How can you kepe up with the changes in handling the validation of content models from the processing of the business logic that triggers and implements business process level services?
Forces	You want to do content validation yet the main purpose is to support business processing through services.
Solution	Separate out the validation of the content model from the actual back-end business processing with EJB's or Web Services or interaction with the legacy systems (See Encapsulate Legacy as Services)
Solution Detail	By isolating the management of business rules and their validation from the rest of the processing, the former can be changed without impacting the next steps in the business processing. This is an extension of the Rule Pattern Language [5].
Resulting Context	You now can alter the validation of the content model without affecting the applications related to processing the subsequent steps in the workflow.
Implementation Considerations	Content Validation may take on many forms, you may use Schematron for assertion-based rules, MXL schema for patterns and regular expressions and Java for actual processing of code-level validation. All these may need to be orchestrated into a sequence of validations that may not necessarily use the same technology throughout, due to the individual limitations of the technologies. For example, DTD's cannot define reusable Types or sequences or ranges, or pattern matching; whereas using XML Schema these can be accomplished. Alternatively, XML Schema is data-based and cannot process conditional assertions. For that, a tool like Schematron may be used to handle the assertion based rules for content validation of XML content – e.g., as when it is send through a SOAP for Data Transfer context.
Known Uses	Rule Pattern Language; International Patent Application Submission and Validation

16 Other Patterns

These are patterns alluded to in the context of some patterns but due to space and time limitations have not been incorporated into this paper. These include:

1. Selective Exposure
2. Encapsulate Legacy As Services – Explained in article entitled Web Services: Promises and Compromises, forthcoming, IEEE Outlook.

17 References

1. Web Services Architecture Working Group
2. Gamma, Helm, Johnson, Vlissides, Design Patterns, Elements of Reusable Object-oriented Software, Addison-Wesley, 1994.
3. Web Services Invocation Framework; Java Community Process API, JCP.
4. Web Services Gateway; www.alphaworks.ibm.com
5. Arsanjani, A., Rule Pattern Language, Pattern Languages of Programming 1999.